DW_asymdata_outbuf

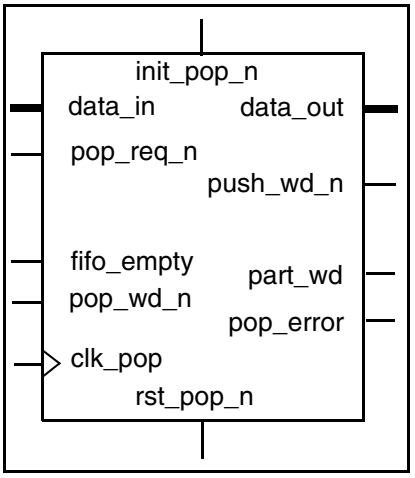
Asymmetric Data Output Buffer

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- Parameterized asymmetric input and output data widths (*in_width* > *out_width* with integer multiple relationship)
- Parameterized byte (sub-word) ordering within a word
- Parameterized flush value
- Word integrity flag
- Parameterized error status mode
- Registered push error (overflow)



Description

This component multiplexes input data stream words and into output sub-words predicated on the input data width being greater than and an integer multiple of the output data width. Once all of sub-words from the full word data input are presented to the data output, a pop word enable signal is asserted.

This component was initially conceived as a back-end piece to an asymmetric First-In-First-Out (FIFO) device as depicted in the data flow usage example shown in Figure 1-2 on page 5. Note that the interface naming of DW_asymdata_outbuf incorporates the "pop" nomenclature associated with a FIFO device.

A "flush" feature is available to force out a partially buffered word. This functionality is useful in clearing the input buffers and establishing word alignment.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk_pop	1	Input	Clock source
rst_pop_n	1	Input	Asynchronous reset (active low)
init_pop_n	1	Input	Synchronous reset (active low)
pop_req_n	1	Input	Pop request (active low)
data_in	*in_width*	Input	Input data (sub-word)
fifo_empty	1	Input	Empty indication connected RAM/FIFO
pop_wd_n	1	Output	Full data word transferred (active low)
data_out	*out_width*	Output	Output data (sub-word)

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
part_wd	1	Output	Partial word popped flag
pop_error	1	Output	Under-run of RAM/FIFO

Table 1-2 Parameter Description

Parameter	Values	Description
in_width	1 to 2048 Default: 8	Width of `data_in` Must be greater than *out_width* and an integer multiple; that is, *in_width* = K * *out_width*
out_width	1 to 2048 Default: 16	Width of `data_out` Must be less than *in_width* and an integer multiple; that is, *in_width* = K * *out_width*
err_mode	0 or 1 Default: 0	Error flag behavior mode ■ 0: Sticky `pop_error` flag (hold on first occurrence, clears only on reset) ■ 1: Dynamic `pop_error` flag (reports every occurrence of error)
byte_order	0 or 1 Default: 0	Sub-word ordering into Word 0: The first byte (or sub-word) is in MSB of word 1: The first byte (or sub-word) is in LSB of word

Table 1-3 Synthesis Implementations

Implementation Name	Function	License Feature Required
rtl	Synthesis model	DesignWare

Table 1-4 Simulation Models

Model	Function
DW03.DW_ASYMDATA_OUTBUF_CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_asymdata_outbuf_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_asymdata_outbuf.v	Verilog simulation model source code

Figure 1-1 is a block diagram of the DW_asymdata_outbuf component.

Figure 1-1 DW_asymdata_outbuf Basic Block Diagram

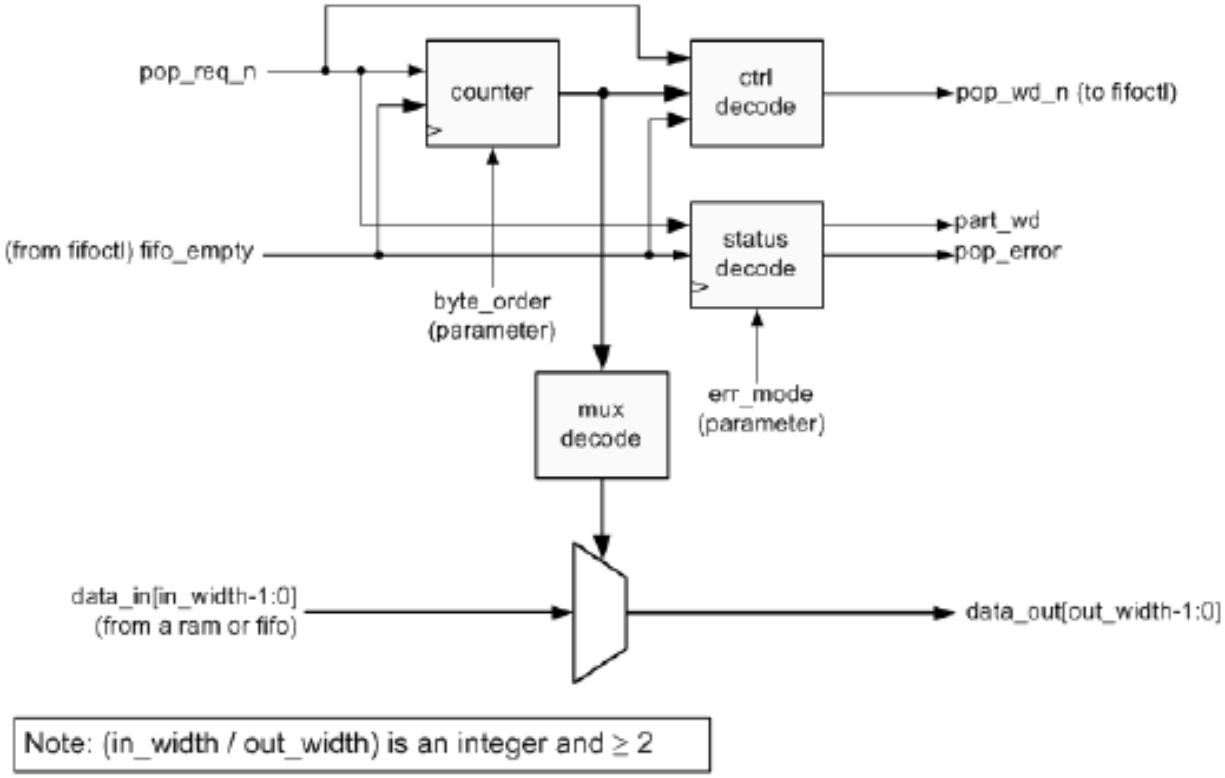
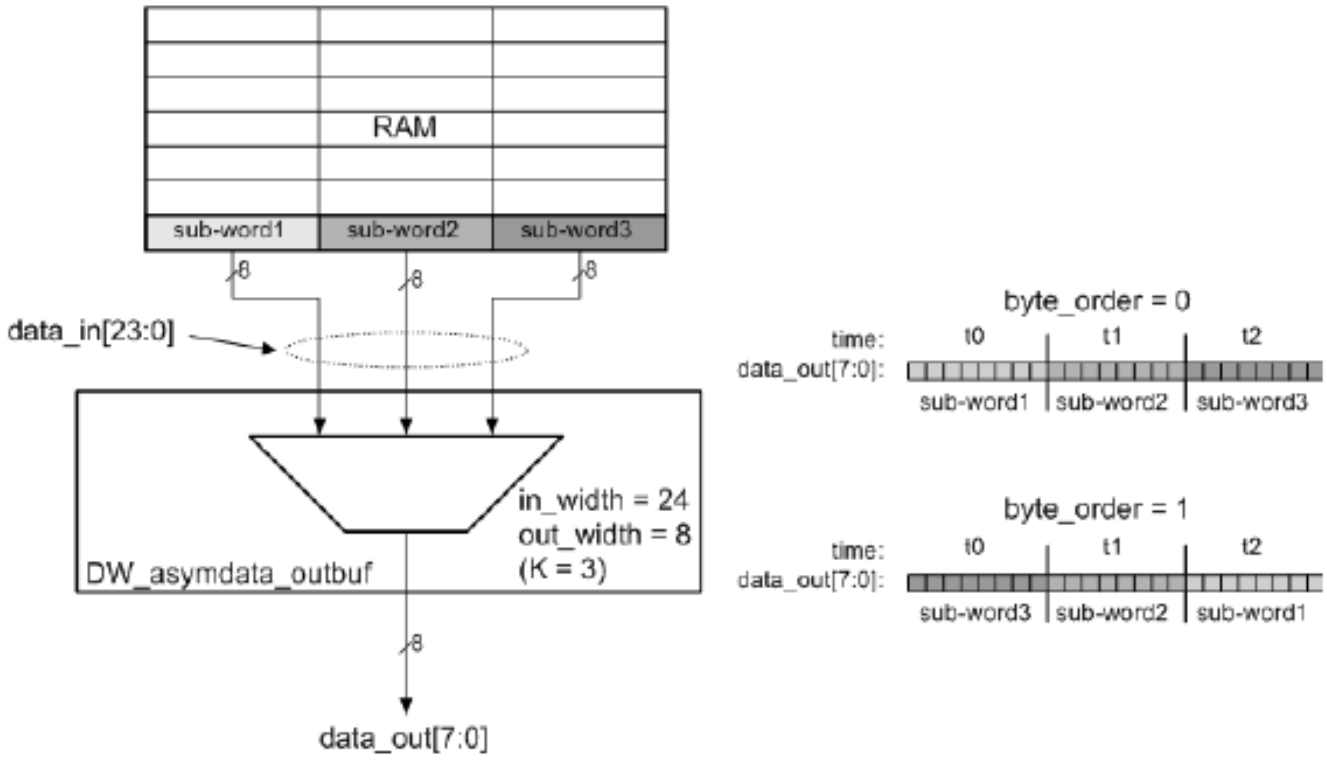


Figure 1-2 on page 4 shows how the output multiplexer routes `data_in` to `data_out` (based on the value of *byte_order*) from a RAM element.

Figure 1-2 Data Flow Based on 'byte_order' Value



Partial Words

When the sub-words are being multiplexed to `data_out` before the full word of `data_in` is traversed, the output flag `part_wd` is active (high). After K pops, where K = *in_width* / *out_width*, K sub-words are presented to the `data_out` output. When all sub-words for a particular word from the RAM/FIFO are sent out, `part_wd` goes inactive (low) on the following cycle. The order of how the sub-words are retrieved from a word is determined by the *byte_order* parameter (as shown in Figure 1-2).

Popping Complete Words

As the Kth sub-word is being popped (`pop_req_n` asserted low) to `data_out`, the `data_in` complete word has then been traversed, then `pop_wd_n` asserts for a single `clk_pop` cycle. The `pop_wd_n` output is a non-registered single-clock pulse and is a combinational result derived directly from `pop_req_n`. So, there must be an awareness of the timing characteristics of `pop_req_n`.

Pop Errors

A pop error occurs if `pop_req_n` asserts during active `fifo_empty`. The `pop_error` output is registered and its behavior can either be 'sticky' or 'dynamic' based on the *err_mode* parameter. See Table 1-2 on page 2 for the *err_mode* definition.

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

Timing Waveforms

Figure 1-3 depicts, after an asynchronous reset, a sequence that pops a complete word. The data widths configurations are *in_width* of 24 and *out_width* of 8 which makes the integer multiple of *in_width* to *out_width*, K, equal to 3. On the third active `pop_req_n`, `pop_wd_n` goes active (low) indicating that a complete word have been popped. The *byte_order* parameter in this case is set to 1, which means that the first sub-word popped is taken from the least significant sub-word of `data_in`.

Figure 1-3 Pop a Complete Word (Parameter Set 1)

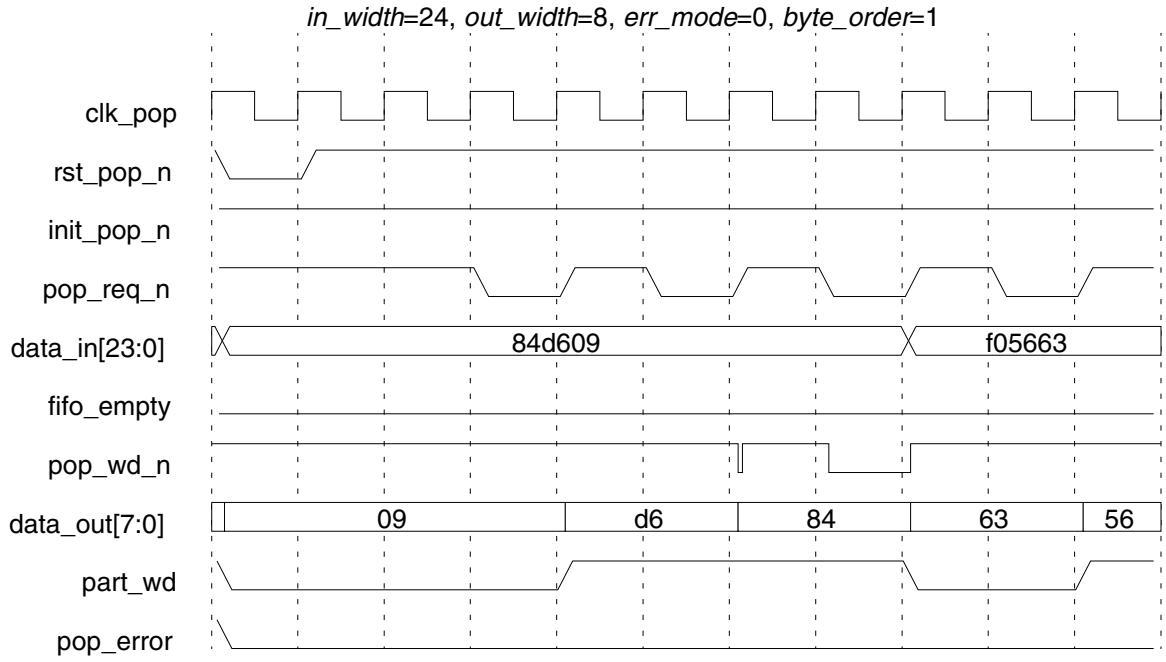
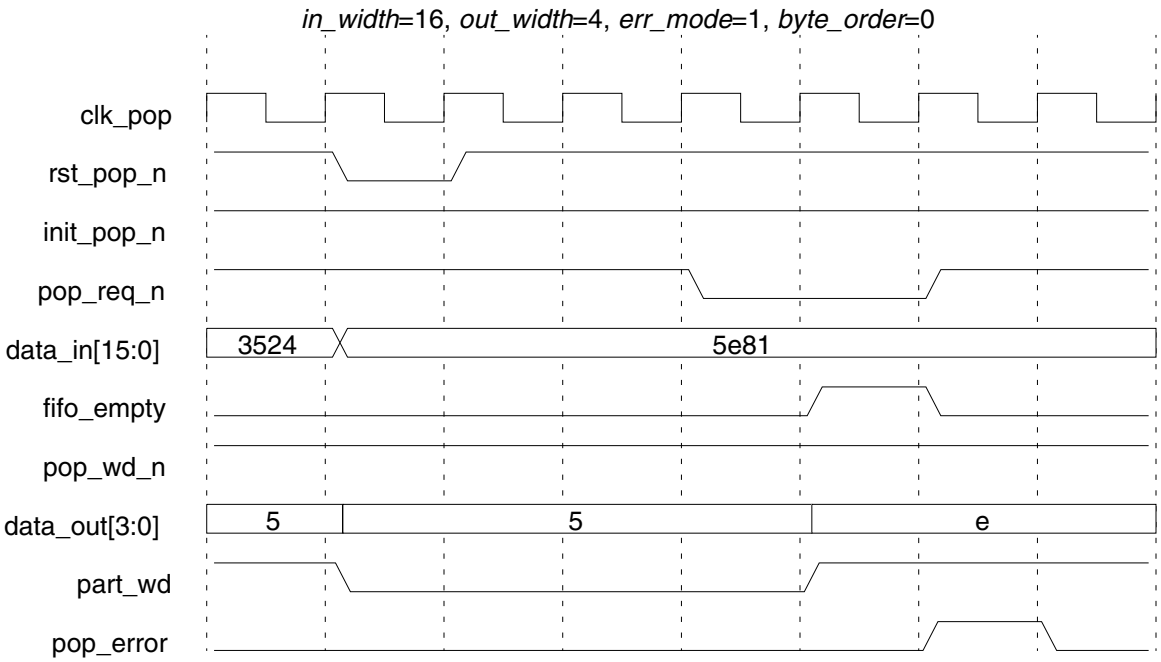


Figure 1-4 shows a pop error condition. The `pop_error` signal is asserted on the next clock cycle after `pop_req_n` and `fifo_empty` are asserted. The *err_mode* parameter is set to 1, which means the `pop_error` assertion only lasts as longer as the error condition is present. Thus, after `fifo_empty` de-asserts, `pop_error` also de-asserts on the next cycle. If *err_mode* was 0, `pop_error` would have remained asserted until a reset to the component was initiated.

Figure 1-4 Pop a Complete Word and Error (Parameter Set 2)



Related Topics

- Memory – FIFO Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_asymdata_outbuf_inst is
      generic (
        inst_in_width : INTEGER := 16;
        inst_out_width : INTEGER := 8;
        inst_err_mode : INTEGER := 0;
        inst_byte_order : INTEGER := 0
        );
      port (
        inst_clk_pop : in std_logic;
        inst_rst_pop_n : in std_logic;
        inst_init_pop_n : in std_logic;
        inst_pop_req_n : in std_logic;
        inst_data_in : in std_logic_vector(inst_in_width-1 downto 0);
        inst_fifo_empty : in std_logic;
        pop_wd_n_inst : out std_logic;
        data_out_inst : out std_logic_vector(inst_out_width-1 downto 0);
        part_wd_inst : out std_logic;
        pop_error_inst : out std_logic
        );
    end DW_asymdata_outbuf_inst;


architecture inst of DW_asymdata_outbuf_inst is
begin

    -- Instance of DW_asymdata_outbuf
    U1 : DW_asymdata_outbuf
    generic map ( in_width => inst_in_width, out_width => inst_out_width, err_mode =>
inst_err_mode, byte_order => inst_byte_order )
    port map ( clk_pop => inst_clk_pop, rst_pop_n => inst_rst_pop_n, init_pop_n =>
inst_init_pop_n, pop_req_n => inst_pop_req_n, data_in => inst_data_in, fifo_empty =>
inst_fifo_empty, pop_wd_n => pop_wd_n_inst, data_out => data_out_inst, part_wd =>
part_wd_inst, pop_error => pop_error_inst );


end inst;

-- Configuration for use with a VHDL simulator
-- pragma translate_off
library DW03;
configuration DW_asymdata_outbuf_inst_cfg_inst of DW_asymdata_outbuf_inst is
  for inst
  end for; -- inst
```

```
end DW_asymdata_outbuf_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_asymdata_outbuf_inst( inst_clk_pop, inst_rst_pop_n,
    inst_init_pop_n, inst_pop_req_n, inst_data_in, inst_fifo_empty,
    pop_wd_n_inst, data_out_inst, part_wd_inst, pop_error_inst );

parameter inst_in_width = 16;
parameter inst_out_width = 8;
parameter inst_err_mode = 0;
parameter inst_byte_order = 0;


input inst_clk_pop;
input inst_rst_pop_n;
input inst_init_pop_n;
input inst_pop_req_n;
input [inst_in_width-1 : 0] inst_data_in;
input inst_fifo_empty;
output pop_wd_n_inst;
output [inst_out_width-1 : 0] data_out_inst;
output part_wd_inst;
output pop_error_inst;

    // Instance of DW_asymdata_outbuf
    DW_asymdata_outbuf #(inst_in_width, inst_out_width, inst_err_mode, inst_byte_order)
U1 (
                .clk_pop(inst_clk_pop),
                .rst_pop_n(inst_rst_pop_n),
                .init_pop_n(inst_init_pop_n),
                .pop_req_n(inst_pop_req_n),
                .data_in(inst_data_in),
                .fifo_empty(inst_fifo_empty),
                .pop_wd_n(pop_wd_n_inst),
                .data_out(data_out_inst),
                .part_wd(part_wd_inst),
                .pop_error(pop_error_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
July 2020	DWBB_201912.5	■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 5 and added the DW_SUPPRESS_WARN macro
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section
April 2018	DWBB_201709.5	■ For STAR 9001317257, updated the maximum value for *in_width* and *out_width* parameter in Table 1-2 on page 2 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com